P3: Wrangle OpenStreetMap Data - OpenStreetMap Project

Map Area

Newbury, Berkshire, United Kingdom

I decided to use a custom bounding box as there was not a metro export available for my hometown. The bounding box I used was: - http://www.openstreetmap.org/export#map=10/51.2671/-0.8123 - min Longitude: -1.6658 - min Latitude: 51.2881 - max Longitude: -0.9462 - max Latitude: 51.5702

The area I chose contained the area surrounding my hometown. I am a keen cyclist that uses the OpenStreetMap data to not only plan my cycling routes before a ride but also use them for mapping on my Garmin cycling computer while out on a ride. Having the opportunity to improve the local mapping would not only benefit the community but also my route planning and navigation at the same time.

Problems Encountered in the Map

Having downloaded a sufficient data set to cover the area around Newbury I was pleasantly surprised to find the data was of a much higher quality than I expected. During the audit I chose to focus on auditing and cleaning the cities and steet names. While auditing the cities and street names I was able to generate an expected list and also a mapping list to clean and incorrect or over abbreviated values

Adding to expected steet names list

After reviewing the list of exceptions beyond the initial list used in earlier lessons, I was able to add multiple values to the expected list. As the area sounding my home town contains numerous small villages, unique road/street names exist such as "Rookery", "Glebe" and "Rise" so these were added to the expected list.

```
expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square",
"Lane", "Road",
            "Trail", "Parkway", "Commons", "Close", "Gardens", "Hill", "Way",
"Park", "Centre",
            "Common", "Crescent", "Fields", "Roundabout", "Row", "Ride", "View",
"Walk",
            "Broadway", "Down", "End", "Grove", "Cornfields", "Eastcourt", "Green",
"Link",
            "Mill", "Newfound", "A339", "Fosbury", "Glebe", "Hailey", "Rookery",
"Smithy", "Parade",
            "Arcade", "Estate", "Mall", "Rise", "Horse", "West", "Mead",
"Approach", "Ashbury", "Brow",
            "Butts", "By-pass", "Chase", "Cottages", "Forbury", "Forest", "Gate",
"Heath", "Lea",
            "Market", "Mews", "Oracle", "Pleasant", "Queensway", "Saye", "Terrace",
"Tilehurst", "Limes"
            ]
```

Creation of a mapping for steet name exceptions

Beyond adding to the expected values, any exceptions to this were added via a mapping table. These predominantly revolved around road, street, and avenue over abbreviations. The was one completely incorrect value of "www.cpva.org.uk" which I have updated to "Unknown" in the mappings below

```
mapping = {
            "Rd" : "Road",
            "Rd," : "Road",
            "Road," : "Road",
            "Steet" : "Street",
            "Road," : "Road",
            "Steet" : "Street",
            "Ave" : "Avenue",
            "Sr" : "Street",
            "street" : "Street",
            "www.cpva.org.uk" : "Unknown"
            }
```

Adding to expected cities list

After auditing the cities, I added the below values to the cities_expected list.

```
cities_expected = ["Aldermaston", "Andover", "Ash", "Avington", "Basildon",
"Basingstoke", "Bedwyn",
                   "Bradfield", "Bramley", "Caversham", "Chaddleworth",
"Checkendon", "Curridge", "Goring",
                   "Hook", "Hungerford", "Ilsley", "Inkpen", "Ipsden",
"Kingsclere", "Kintbury",
                   "Mapledurham", "Marlborough", "Midgham", "Mortimer", "Newbury",
"Norreys", "Overton",
                   "Pangbourne", "Shinfield", "Streatley-On-Thames", "Swindon",
"Theale", "Common",
                   "Hill", "Row", "Tadley"
                   ]
```

Creation of a mapping for city exceptions

There were very few incorrect values within the cities, there were a few values that needed to be corrected. I also noticed 4 instances where there were more than one one city value present. In those cases I corrected these to the best of my knowledge.

```
city_mapping = {
```

```
            "READING" : "Reading",
            "THATCHAM" : "Thatcham",
            "Rotherfield Greys / Henley-on-Thames" : "Henley-on-Thames",
            "Caversham, Reading" : "Reading",
            "Lower Basildon, Reading" : "Reading",
            "Pangbourne, Reading" : "Reading"
            }
```

Sort cities by count, descending

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM node_tags UNION ALL
      SELECT * FROM way_tags) tags
WHERE tags.key LIKE '%city'
GROUP BY tags.value
ORDER BY count DESC;
```

Within the OSM extract there were 98 different city tag values present. I have included the top 10 results which have been edited for readability:

```
value       count
Reading     3285
Newbury     274
Basingstoke 121
Tadley      16
4           11
Shrivenham  10
Wantage     9
6           7
10          6
Didcot      6
```

The above results show the most common city value used is Reading, this would be in line with expectations as it is the largest town within my bounding box. It encompasses the eastern portion of the bounding box I selected. The second most common value is Newbury which is my hometown. It is proportionally much smaller than Reading so I would expect nothing less than the count of value to be much smaller.

Data Overview and Additional Ideas

File Sizes

```
Newbury Area OSM 31102016.osm .. 117 MB
OSM Project.db ................. 63.4 MB
nodes.csv ...................... 42.6 MB
nodes_tags.csv ................. 2.23 MB
ways.csv ....................... 4.05 MB
ways_tags.csv .................. 7.64 MB
ways_nodes.cv .................. 15.3 MB
```

Number of Nodes

```
SELECT COUNT(*) FROM node;
```

542,652

Number of Ways

```
SELECT COUNT(*) FROM way;
```

71,444

Number of Unique Users

```
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM node UNION ALL SELECT uid FROM way) e;
```

891

Top 10 Contributing Users

```
SELECT e.user, COUNT(*) as num
FROM (SELECT user FROM node UNION ALL SELECT user FROM way) e
GROUP BY e.user
ORDER BY num DESC
LIMIT 10;
```

```
user                    num
jpennycook              50663
Mark_S                  50056
Eriks Zelenka           46357
ndm                     40164
The Maarssen Mapper     28903
richardwest             25815
Kabads                  25563
GordonFS                21313
Philip                  21091
DanGregory              18799
```

Number of users appearing only once (having 1 post)

```
SELECT COUNT(*)
FROM
    (SELECT e.user, COUNT(*) as num
     FROM (SELECT user FROM node UNION ALL SELECT user FROM way) e
     GROUP BY e.user
     HAVING num=1)  u;
```

144

Top 10 appearing amenities

```
SELECT value, COUNT(*) as num
FROM node_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

```
value              num
post_box           521
bench              461
pub                235
telephone          172
place_of_worship   160
parking            125
bicycle_parking    97
post_office        66
shop               66
emergency_phone    64
```

Top restaurant cuisines in the area

```
SELECT node_tags.value, COUNT(*) as num
FROM node_tags     JOIN (SELECT DISTINCT(id) FROM node_tags WHERE
value='restaurant') i
    ON node_tags.id=i.id
WHERE node_tags.key='cuisine'
GROUP BY node_tags.value
ORDER BY num DESC;
```

```
value              num
indian             10
chinese            6
pizza              4
burger             2
italian            2
thai               2
asian              1
fish_and_chips     1
french             1
international      1
pasta;pizza        1
portuguese         1
regional           1
spanish            1
```

```
steak_house       1
```

Additional Data Exploration

Based on being a cyclist, I thought it would be interesting to see if I could investigate some cycling related information from the database I have created.

Number of bicycle stands in the map area

```
SELECT node_tags.value, COUNT(*) as num
FROM node_tags
    JOIN (SELECT DISTINCT(id) FROM node_tags WHERE value='stands') i
    ON node_tags.id=i.id
WHERE node_tags.key='bicycle_parking'
GROUP BY node_tags.value
ORDER BY num DESC;
```

24

Number of designated bicycle nodes in the map area

```
SELECT node_tags.value, COUNT(*) as num
FROM node_tags
    JOIN (SELECT DISTINCT(id) FROM node_tags WHERE value='designated') i
    ON node_tags.id=i.id
WHERE node_tags.key='bicycle'
GROUP BY node_tags.value
ORDER BY num DESC;
```

23

Benefits and Anticpated Problems with implementing the data improvements

Benefits

1. Owning the data: By implementing our fixes we are taking ownership of insuring the data is correct and also auditing other users updates/additions.
2. Frequency of updates/improvements: When we implement our data improvements we can also govern the frequency at which these are made. If we were a regular contributor we could insure that our local area stays up to date as far as improving the data that other users add.
3. Improvements to freely available data: As I am a cyclist, having access to free correct data would benefit me personally for use of my GPS cycling computer. Having the ability to download would not only benefit me but other cyclists in the area who also use the Open Street Map Data.

Anticipated Problems

1. Incorrect fixes: While improving the data is a great initiative, my opinion on an improvement

might actually end up degrading the data. One person's judgement on possible incorrect data could actually turn out to be a incorrect fix. I suppose, to alleviate this problem, restricting the working area for imrovements to the immediate local area would reduce the risk of incorrect fixes being applied by using local knowledge.

2. Coding errors: While working with large volumes of data there is always the risk that incorrect code could inadvertently introduce problems into the data that had not already existed. An example of this would be replacing short character strings, if a too loose replace is performed it could update valid data to incorrect data very easily. To rectify this, it would be in the best interest to have all fixes validated or cross checked by another user before anything went live.

Conclusion

After reviewing the area surrounding my home town of Newbury, I believe the OpenStreetMap data is far more complete than I first expected. The level of detail the data offers offer allows me to drill down to a very low level, taking the number of bicycle stands as an example. Once the data had been audited and fixed from a street/road perspective, I would be confident to use the data on my GPS device and not being overly concerned that there would be any large errors that could cause me a problem. While reviewing the top 10 contributing users it's clear that there are a lots of very active users contributing to quality of the data in my local area. Every one of the top 10 users have made over 10,000 contributions. While the data cleaning I have completed can contribute to quality of the data, I think it would be a long time till I could become a top contributor. With that in mind programmatically contributing would be a good start to contribute to my local areas data.